

Цель работы

Написать функцию, которая возвращает кортеж из двух индексов элементов списка `lst`, таких что сумма элементов по этим индексам равна переменной `target`.

Элемент по индексу может быть выбран лишь единожды, значения в списке могут повторяться.

Если в списке встречается больше чем два индекса, подходящих под условие - вернуть наименьшие из всех. Элементы находятся в списке в произвольном порядке.

Комментарии по выполнению

[Стартовый борд в repl.it](#)

Для входного набора

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
target = 8
```

программа должна вернуть

```
(0, 6)
```

Для начала решим эту задачу "в лоб" с помощью циклов

```
def two_sum(lst, target):
    pass
```

Усложнение 1

Решим задачу другим способом, чтобы вычислительная сложность решения была не $O(n^2)$, а ниже (например, $O(n)$).

Решение оформить в виде отдельной функции:

```
def two_sum_hashed(lst, target):
    pass
```

Усложнение 2

Усложним задачу и вернем все наборы индексов, удовлетворяющих условию.

Для входного набора:

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
target = 8
```

программа должна вернуть:

```
[(0,6), (1,5), (2,4)]
```

Опубликуйте решение с помощью сервиса repl.it и предоставьте в конце занятия ссылку на собственное решение в repl.it

```
def main():
    lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    target = 8
    print("First result is:")
    two_sum(lst, target)
    two_sum_hashed(lst, target)

def two_sum(lst, target):
    for i in range(len(lst) - 1):
        for j in range(i + 1, len(lst)):
            if lst[j] == target - lst[i]:
                print("{0},{1}".format(i, j))

def two_sum_hashed(lst, target):
    listF = {}
    for k in range(len(lst)):
        listF[k] = lst[k]
    print("\nDictionary of original list:")
    print("{0}".format(listF))
    newList = {}
    for k in range(len(lst)):
        newList[k] = target - listF[k]
    print("\nDictionary of difference between values and target:")
    print("{0}".format(newList))
    slovar = {}
    for k in range(len(lst)):
        keys = list(listF.keys())
        values = list(listF.values())
        num = newList.get(k)
        if num in values[k + 1:len(lst)]:
            num = values.index(num, k + 1, len(lst))
            keyValue = keys.pop(num)
            slovar.setdefault(k, [keyValue])
    print("\nItems of result dictionary:")
    print("{0}".format(slovar.items()))

main()
# + 1 формирование словаря из листа
# + 2 вычисление для каждого элемента листа его разницы diff с target
# + 3 поиск в словаре значения разницы
# + 4 проверить не совпадают ли индексы двух найденных элементов на шаге 2 и 3
```

Подсказка: для шага 4 хорошо использовать методы keys или values или items, get у объекта типа dict.

Усложнение 1 Решим задачу другим способом, чтобы вычислительная сложность решения была не $O(n^2)$, а ниже (например, $O(n)$).

Усложнение 2 Усложним задачу и вернем все наборы индексов, удовлетворяющих условию. Программа должна вернуть: [(0,6), (1,5), (2,4)]

<https://repl.it/@IlyaShumyakin/Lab4#main.py>

The screenshot shows a web-based Python IDE interface. The browser address bar displays `repl.it/@IlyaShumyakin/Lab4#main.py`. The interface includes a file explorer on the left with `main.py` selected, a central code editor, and a console on the right.

The code in `main.py` defines two functions: `two_sum` and `two_sum_hashed`. `two_sum` uses a nested loop to find pairs of indices that sum to a target. `two_sum_hashed` uses a hash map (dictionary) to achieve a linear time complexity. It creates a dictionary of the original list, then iterates through it to find complementary values in a second list, storing the results in a dictionary.

```
7
8
9 def two_sum(lst, target):
10     for i in range(len(lst) - 1):
11         for j in range(i + 1, len(lst)):
12             if lst[j] == target - lst[i]:
13                 print("{} {}".format(i, j))
14
15
16 def two_sum_hashed(lst, target):
17     listF = {}
18     for k in range(len(lst)):
19         listF[k] = lst[k]
20     print("\nDictionary of original list:")
21     print("{} {}".format(listF))
22     newList = {}
23     for k in range(len(lst)):
24         newList[k] = target - listF[k]
25     print("\nDictionary of difference between values and target:")
26     print("{} {}".format(newList))
27     slovar = {}
28     for k in range(len(lst)):
29         keys = list(listF.keys())
30         values = list(listF.values())
31         num = newList.get(k)
32         if num in values[k + 1:len(lst)]:
33             num = values.index(num, k + 1, len(lst))
34             keyValue = keys.pop(num)
35             slovar.setdefault(k, [keyValue])
36     print("\nItems of result dictionary:")
37     print("{} {}".format(slovar.items()))
38
39
40 main()
```

The console on the right shows the output of the program, which is currently empty, indicating that the code has not yet been executed.

IlyaShumyakin / Lab_4

Run

Press **F11** to exit full screen

UpgradeShare+

Files

main.py

```
main.py
28 for k in range(len(lst)):
29     keys = list(listF.keys())
30     values = list(listF.values())
31     num = newList.get(k)
32     if num in values[k + 1:len(lst)]:
33         num = values.index(num, k + 1, len(lst))
34         keyValue = keys.pop(num)
35         slovar.setdefault(k, [keyValue])
36 print("\nItems of result dictionary:")
37 print("{}".format(slovar.items()))
38
39
40 main()
41 # + 1 формирование словаря из листа
42 # + 2 вычисление для каждого элемента листа его разницы diff с target
43 # + 3 поиск в словаре значения разницы
44 # + 4 проверить не совпадают ли индексы двух найденных элементов на
шаге 2 и 3
45
46 # Подсказка: для шага 4 хорошо использовать методы keys или values
или items, get у объекта типа dict.
47
48 # Условие 1 Решив задачу другим способом, чтобы вычислительная
сложность решения была не O(n^2), а ниже (например, O(n)).
49 # Условие 2 Условим задачу и вернем все наборы индексов,
удовлетворяющих условию. Программа должна вернуть: [(0,6), (1,5), (2,
4)]
50
```

Console

Shell

Python 3.8.2 (default, Feb 26 2020, 02:56:10)
➤